USING COLLECTIVE INTELLIGENCE TO ROUTE INTERNET TRAFFIC

David H. Wolpert
NASA Ames Research Center
Moffett Field, CA 94035
dhw@ptolemy.arc.nasa.gov

Kagan Tumer
NASA Ames Research Center
Moffett Field, CA 94035
kagan@ptolemy.arc.nasa.gov

Jeremy Frank
NASA Ames Research Center
Moffett Field, CA 94035
frank@ptolemy.arc.nasa.gov

Abstract

A Collective Intelligence (COIN) is a community of interacting reinforcement learning (RL) algorithms designed so that their collective behavior maximizes a global utility function. We introduce the theory of COINs, then present experiments using that theory to design COINs to control internet traffic routing. These experiments indicate that COINs outperform previous RL-based systems for such routing that have previously been investigated.

1 INTRODUCTION

CHECK *ALL* FORMATTING INSTRUCTIONS. Send this paper to Applications.

COllective INtelligences (COINs) are (usually) very large, sparsely connected recurrent neural nets, whose "neurons" are themselves reinforcement learning (RL) algorithms. The distinguishing feature of a COIN is that its dynamics involves no centralized control, but only the collective effects of the individual neurons each modifying their behavior via their (local) RL algorithms. This restriction holds even though the goal of the COIN concerns the system's global behavior.

One naturally-occuring COIN is a human economy, where the "neurons" consist of individual humans trying to maximize their reward, and the "goal", for example, can be viewed as having the overall system achieve high gross domestic product. The robustness and power of naturally-occuring COINs suggests it would be useful to construct Artificial COINs (ACOINs), for example as controllers of distributed

systems. This paper presents a preliminary investigation of this idea for the problem of distributed routing of internet traffic. See [10] for work on other applications of COINs.

In designing an ACOIN, we start with a global utility function concerning global behavior. Our task is to initialize and then update the neurons' utility functions so that as the neurons improve their utilities, global utility also improves. (We may also wish to update the local topology of the net to the same end.) In particular, we need to ensure that the neurons do not "frustrate" each other as they attempt to increase their utilities. All of this must be done without centralized control.

We refer to the RL algorithms at each neuron as *microlearnering*, and the updating of the ACOIN is known as *macrolearning*. For robustness and breadth of applicability, we assume essentially no knowledge concerning the dynamics of the full system, i.e., the macrolearning and/or microlearning must "learn" that dynamics, implicitly or otherwise. This rules out any approach that models the full system.

The problem of designing an ACOIN has never previously been addressed in full — hence the need for introducing new concepts like those described below. Nonetheless, this problem is related to previous work in many fields: distributed artificial intelligence, multi-agent systems, computational ecologies, adaptive control, computational economics, game theory (and in particular stochastic game theory, evolutionary game theory, and game-theoretic mechanism design), voting and auction theory, computational markets, iterated function systems, active walker models, self-organized criticality, spin glasses, reinforcement learning, recurrent neural nets, Markov decision process theory, network theory, and ant-based optimization.

For the particular problem of routing, the most relevant previous work is that of [3, 8]. Like that previous work, we identify as a separate "neuron" each pair of a router and an ultimate destination in the net. That previous work focusses on using RL to update the internal parameters of neurons implementing conventional shortest path algorithms (SPAs), either using centralized control or (more usually) ignoring the problem of neurons working at cross purposes. It does not try to design a full-fledged COIN.

We compared three algorithms for internet routing using numerical simulations. The first two are an SPA and an ACOIN where each neuron knows i) the quickest possible path to the ultimate destination, for SPA's; ii) the reward signal-maximizing path, for COINs. The third algorithm was an ACOIN using a memory-based microlearner [1] having limited knowledge.

The performance of the perfect-knowledge ACOIN was the theoretical optimum. The performance of the perfect-knowledge SPA was PERCENT of this optimum. Despite limited knowledge, the memory-based ACOIN outperformed the perfect knowledge SPA, reducing the deficit in that algorithm's performace by PERCENT. Note that the performace of the perfect knowledge SPA is an upper bound on the performance of any RL-based SPA.

The next section presents a cursory overview of the mathematics behind ACOINs. The following section presents computer experiments concerning that mathematics, experiments that establish its power in the context of routing problems. That final section presents conclusions and summarizes future research directions.

2 MATHEMATICS OF COINS

The mathematical framework for COINs in general is quite extensive [9]. This paper concentrates on four of the concepts from that framework: subworlds, factored systems, constraint-alignment, and the wonderful-life utility function.

i) We consider the state of the system across a set of $T+1$ consecutive timesteps, which we write without loss of generality as $t \in [0, ..., T]$. Let $\underline{\zeta}_{\eta,t}$ be the (Euclidean-vector-valued) state of neuron η at time t. World utility is a function of the state of all neurons across all t, which we write as $G(\underline{\zeta})$.

ii) To simplify the design of the ACOIN, we group neurons into *subworlds*, all sharing the same *subworld utility*. In the ACOINs considered here, subworlds have no overlap, and a neuron's microlearner receives the utility of that neuron's subworld as its reward signal. We indicate the state of neuron σ in subworld ω at time t by $\underline{\zeta}_{\omega,\sigma,t}$, and by $g_\omega(\underline{\zeta})$ we indicate the subworld utility of subworld ω.

iii) One of the primary desiderata in designing an ACOIN is to have it be *subworld-factored*. This means that a change at time 0 to the state of the neurons in subworld ω results in an increased value for $g_\omega(\underline{\zeta})$ iff it results in an increased value for $G(\underline{\zeta})$.

For a subworld-factored system, ω's increasing its own utility does not have "side-effects" on the rest of the system that decrease world utility. For these systems, the separate neurons pursuing their separate goals do not frustrate each other, as far as world utility is concerned. In particular, consider a Nash equilibrium of the system, i.e., a state in which no neuron can increase its utility by changing its action, given the actions of the other neurons [6, 7]. The following is proven in [us]:

Theorem 1: If a system is subworld-factored, then any state at time 0 of the system that constitutes a Nash equilibrium for the subworld utilites is a critical point of world utility. Conversely, a maximum of world utility is such a Nash equilibrium.

In other words, for a subworld-factored system, although it may be maximized at other points as well, world utility is assuredly maximized when each of the neurons' reinforcement learning algorithms are performing as well as they can, given each others' behavior. So we have the correct behavior if and when the learners reach an equilibrium. There can be no tragedy of the commons [4], etc.

iv) A (perfectly) *constraint-aligned* system is one in which any change to the state of the neurons in subworld ω at time 0 will have no effects on the states of neurons outside of ω at times later than 0.

v) Let $Zero_\omega(\underline{\zeta})$ be defined as the vector $\underline{\zeta}$ modified by setting the values of all neurons in subworld ω across all time to $\overline{0}$. The *wonderful life subworld utility* (WLU) is $g_\omega(\underline{\zeta}) \equiv G(\underline{\zeta}) - G(Zero_\omega(\underline{\zeta}))$.

Intuitively, the WLU is analogous to the change in world utility that would have arisen if subworld ω had never existed. (Hence its name - cf. the Frank Capra movie.) Note though that no assumption is made that $Zero_\omega(\underline{\zeta})$ is consistent with the dynamics of the system. In particular, to evaluate the WLU we do *not* have to infer how the system would have evolved if all neurons in ω were set to 0 at time 0. So long as we know $\underline{\zeta}$ (by observation, in the process of macrolearning that occurs at time T), and so long as we know G (set in the problem specification of the ACOIN), we can evaluate the WLU. This is true even if we nothing of the dynamics of the system. This dynamics-independence is a major strength of the WLU.

In addition to assuring the correct equilibrium behavior, there exist many other

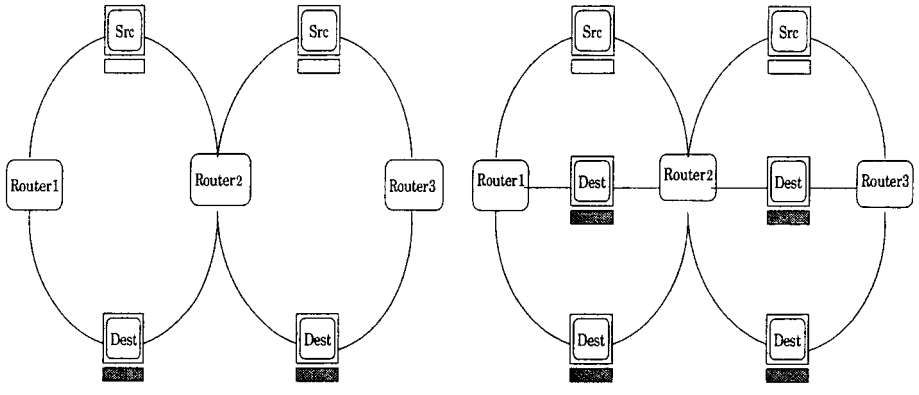
theoretical advantages to having a system be subworld-factored [us]. In this paper we use computer experiments to establish the applicability of these advantages to network routing. In particular, the following theorem, proved in [us], serves as the basis for the investigation of this paper:

Theorem 2: A constraint-aligned system with wonderful life subworld utilities is subworld-factored.

3 EXPERIMENTS

3.1 Network Architectures and mapping to COINs

In our experiments we concentrated on the two network architectures displayed in Figure 1 (note that both are slightly larger than those in [5]). To facilitate the analysis, traffic originated only at routers indicated (SRC) , and had only the other routers indicated (DEST) as ultimate destinations. Although each router can in principle route traffic, only those that are neither originators, nor ultimate destinations were designated as "router" in Figure 1. Note that in both networks there is bottleneck in the central router.



(a) Network A (b) Network B

Figure 1: Network Architectures.

As is standard in much of traffic network analysis [2], at any time, all traffic at a router is a real-valued number together with an ultimate destination tag. At each timestep, each router sums all traffic received in this timestep, to get a total *load*. A running average is kept of each router's load over a window of the previous L timesteps, and that average is run through a *load-to-delay* function to get the router's total *delay* at this timestep. To insure a heterogenous pool of routers, different routers had different load-to-delay functions. This situation more accurately reflects a real network where differences in router software and hardware (response time, queue length, processing speed etc.) affect the resulting delays. In the experiments reported here, the load-to-delay functions were $W(\cdot) = x^3$ for routers 1 and 3, and $W(\cdot) = log(x + 1)$ for router 2, for both networks.

In terms of the COIN formalism, we identified the neurons η as individual pairs of routers and ultimate destinations. So $\underline{\zeta}_{\eta,t}$ was the vector of traffic sent along all

links exiting η's router, tagged for η's ultimate destination, at time t. Each subworld consisted of the set all neurons that shared a particular ultimate destination.

An SPA tries to set $\underline{\zeta}_{\eta,t}$ to minimize the sum of the total delays along all possible paths to η's ultimate destination. Note the greedy nature of this scheme. An ACOIN instead tries to set $\underline{\zeta}_{\eta,t}$ to maximize g_ω for the subworld ω containing ζ. For both algorithms, "perfect knowledge" means that at time t, all of the routers know the values of $\underline{\zeta}$ for time $t-1$, and assume that those values will be the same at t. For large enough window sizes L, this assumption will be arbitrarily good in most cases, and therefore will allow the routers to make arbitrarily accurate estimates of how best to route their traffic, according to their respective routing criteria.

The load at router r at time t is determined by $\underline{\zeta}$. Accordingly, we can encapsulate the load-to-delay functions at the nodes by writing the delay at node r at time t as $W_{r,t}(\underline{\zeta})$. In our experiments world utility was the summed-total-delay, i.e., $G(\underline{\zeta}) = \sum_{r,t} W_{r,t}(\underline{\zeta})$. So using the WLU, $g_\omega(\underline{\zeta}) = \sum_{r,t}[W_{r,t}(\underline{\zeta}) - W_{r,t}(Zero_\omega(\underline{\zeta}))]$. Now $W_{r,t}(Zero_\omega(\underline{\zeta}))$ is the delay that would have existed at router r at time t if all the traffic from r at t that was destined for ω had not existed. So for our G, the WLU is just a sum of differences in delays "caused" by the routing of the neurons in subworld ω.

The evaluation of this subworld utility in the memory-based-reasoning COIN was trivial to implement. All traffic had a header containing a running sum of the differences in total delays encountered in all the routers it has traversed so far. Then each ultimate destination summed all such headers it received and echoed that sum back to all routers that had routed to it.

In this way each neuron is apprised of the subworld utility of its subworld that resulted (in part) from its routing decisions. The memory-based microlearner worked by forming a training set where load distribution decisions were mapped to wonderful life utility functions. More precisely, for each router-ultimate destination pair, the training patterns had the loads on each outgoing link as inputs, and the resulting woderful life utilities as outputs. Then the wonderful life utility function corresponding to each routing decision was estimated by using a nearest neighbor algorithm, and the router sent the packets along the path that would improve its wonderful life utilty function.

3.2 Simulation Results

The networks discussed above tested under light, medium and heaviy traffic. For simulation purposes traffic density corresponded to the number of source-destination pairs that are active at any one time. For heavy traffic, all sources were sending packets to all destinations at each time step; for light traffic, each source only sent packets to half the destinations; and for medium traffic, on the average each source sent packets to 75% of the destinations.

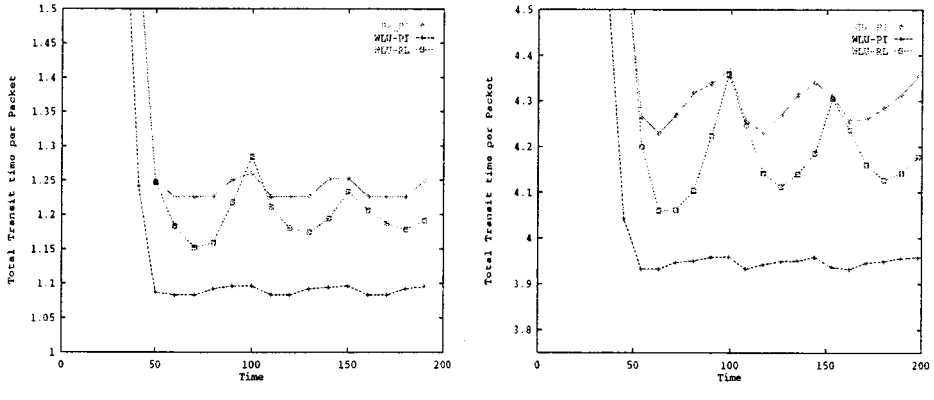
Table 1 reports the average traversal times per individual packet in each of the traffic regimes, for the shortest path algorithm with perfect knowledge, the ACOIN with perfect knowledge and the ACOIN with memory-based learning. Each entry is based on 50 runs and the differences are all statistically significant, including SPA-PI vs. ACOIN-RL for Network A under light traffic conditions (i.e., the hypothesis that the two means are equal is rejected at the $\alpha = 0.01$ level for the t-test.)

Table 1 provides two important observations: First, the wonderful life utility based ACOIN by outperforming the SPA when both have perfect knowledge demonstrates the superiority of the new routing mechanism. By not greedily pursuing the quickest

Table 1: Total per Packet Transit Times

Network	Load	SPA-PI	ACOIN-PI	ACOIN-RL
A	light	0.53 ± .05	0.45 ± .01	0.50 ± .06
	medium	1.26 ± .07	1.10 ± .01	1.21 ± .06
	heavy	2.17 ± .08	1.93 ± .01	2.06 ± .07
B	light	2.13 ± .08	1.92 ± .01	2.05 ± .08
	medium	4.37 ± .10	3.96 ± .01	4.19 ± .08
	heavy	6.94 ± .11	6.35 ± .01	6.82 ± .17

reward (shortest path), the ACOIN settles into a more desirable state that reduces the average traversal times for *all* packets. Second, even when the wonderful life utility is estimated through a memory based learner (through information available at the local routers), the performance of the ACOIN still surpasses that of the perfect knowledge shortest path algorithm. This result not only establishes the feasibility of ACOIN based routers, but also demonstrates that for this tasks ACOINs will outperform any algorithm that attempts at estimating the path information through reinforcement/memory based learning. This result follows from the observation that the perfect information shortest path algorithm is necessarily a ceiling on any algorithm based on routing tables, since all the delays that otherwise would have be estimated are readily available.



(a) Network A (b) Network B

Figure 2: Total Traversal Times.

4 DISCUSSION

Acknowledgements

References

[1] C. G. Atkenson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, Submitted, 1996.

[2] D. Bertsekas and R. Gallager. *Data Networks.* Prentice Hall, Englewood Cliffs, NJ, 1992.

[3] J. Boyan and M. Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. In J. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems - 6*, pages 671–678. Morgan Kaufmann, 1994.

[4] B. L. Crowe. The tragedy of the commons revisited. *Science*, 166:1103–1107, 1969.

[5] P. Marbach, O. Mihatsch, M. Schulte, and J. Tsisiklis. Admission control and routing in integrated service networks. In *Advances in Neural Information Processing Systems - 10.* Morgan Kaufmann, To Appear, 1998.

[6] J. F. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences of the United States of America*, 36(48-49), 1950.

[7] M. Osborne and A. Rubenstein. *A Course in Game Theory.* MIT Press, Cambridge, MA, 1994.

[8] D. Subramanian, P. Druschel, and J. Chen. Ants and reinforcement learning: A case study in routing in dynamic networks. *Proceedings of the 15th International Conference on Artificial Intelligence*, pages 832–838, 1997.

[9] D Wolpert, J. Frank, K. Tumer, and K. Wheeler. Coins. *NOWHERE*, SUBMITTED 1998.

[10] D. Wolpert, K. Wheeler, and K. Tumer. Coins for bars. In *Advances in Neural Information Processing Systems - 11.* Morgan Kaufmann, SUBMITTED 1998.